**SQL Assignment 1**

1. What is a relational database management system (RDBMS)? What are the advantages of a database management system over a file system?

A relational database management system (RDBMS) is a software application that organizes, stores, and retrieves data in a structured manner. It is designed to handle large amounts of data efficiently and effectively. RDBMSs are widely used in a variety of applications, including business intelligence, customer relationship management (CRM), and e-commerce.

**Advantages of a database management system (DBMS) over a file system:**

* **Data integrity:** DBMSs enforce data integrity by ensuring that data is accurate, consistent, and complete. This is achieved through the use of data validation rules, data constraints, and data redundancy controls.
* **Data security:** DBMSs provide a variety of security features to protect data from unauthorized access, modification, or deletion. These features include user authentication, access control lists, and encryption.
* **Data sharing and concurrency:** DBMSs allow multiple users to access and share data simultaneously. This is achieved through the use of locking mechanisms and concurrency control protocols.
* **Data backup and recovery:** DBMSs provide mechanisms for backing up data and recovering it in case of a system failure or data corruption. This is essential for protecting valuable business data.
* **Data manipulation language (DML):** DBMSs provide a high-level data manipulation language (DML), such as SQL, for manipulating data in a structured and efficient way. This makes it easy for users to query, insert, update, and delete data.
* **Data reporting and analysis:** DBMSs provide tools for generating reports and performing data analysis. This helps users to gain insights from their data and make better decisions.

1. In a database management system, explain the ACID properties.

ACID stands for Atomicity, Consistency, Isolation, and Durability. These four properties are essential for ensuring the integrity and reliability of data in a database management system (DBMS).

* **Atomicity:** Atomicity ensures that a transaction is either completed in its entirety or not at all. This means that either all of the operations within a transaction are successful, or none of them are. This prevents data from being left in an inconsistent state in the event of a system failure or other error.
* **Consistency:** Consistency ensures that a transaction maintains the consistency of the data. This means that the data is in a valid state before and after the transaction is executed. This is achieved through the use of data constraints and data validation rules.
* **Isolation:** Isolation ensures that transactions execute independently of each other. This means that a transaction cannot see the intermediate results of another transaction. This prevents data from being corrupted or overwritten by concurrent transactions.
* **Durability:** Durability ensures that committed transactions are not lost in the event of a system failure or other error. This means that the changes made by a committed transaction are permanently stored in the database. This is achieved through the use of data logging and recovery mechanisms.

1. Explain the concept of normalization.

Normalization is a process of organizing data in a database to minimize redundancy and improve data integrity. It involves dividing larger tables into smaller tables and linking them using relationships. Normalization is important because it can help to:

* **Reduce data redundancy:** Redundancy occurs when the same data is stored in multiple places in the database. This can waste storage space and make it difficult to keep the data consistent. Normalization helps to eliminate redundancy by storing data in a single place and then using relationships to link the data to other tables.
* **Improve data integrity:** Data integrity refers to the accuracy and consistency of data in a database. Normalization can help to improve data integrity by reducing the risk of data anomalies. Data anomalies occur when there are inconsistencies in the data, such as when the same data is stored in multiple places or when there are conflicting relationships between tables.
* **Make databases easier to maintain:** Normalized databases are easier to maintain because the data is organized in a logical and consistent way. This makes it easier to add, update, and delete data without introducing errors.

There are three main levels of normalization:

* **First Normal Form (1NF):** A table is in 1NF if it does not contain any repeating groups. Repeating groups occur when the same data is stored in multiple columns of the table.
* **Second Normal Form (2NF):** A table is in 2NF if it is in 1NF and all of its non-key attributes are fully functionally dependent on the primary key. Fully functional dependent means that each non-key attribute is uniquely determined by the primary key.
* **Third Normal Form (3NF):** A table is in 3NF if it is in 2NF and all of its non-key attributes are transitively dependent on the primary key. Transitively dependent means that a non-key attribute is dependent on another non-key attribute, which in turn is dependent on the primary key.

1. Explain the many types of query languages used in relational databases. DQL, DML, DCL, and DDL are some examples.

**Data Definition Language (DDL)**

DDL is used to define the structure of a database, including creating tables, indexes, and views. It is also used to modify the structure of a database, such as adding or removing columns to a table. Some examples of DDL commands include CREATE, ALTER, and DROP.

**Data Manipulation Language (DML)**

DML is used to manipulate data in a database, such as inserting, updating, and deleting data. It is also used to query data from a database. Some examples of DML commands include INSERT, UPDATE, DELETE, and SELECT.

**Data Control Language (DCL)**

DCL is used to control access to data in a database. It is used to grant and revoke permissions to users and groups. Some examples of DCL commands include GRANT, REVOKE, and CREATE USER.

**Data Query Language (DQL)**

DQL is used to query data from a database. It is the most commonly used type of query language in relational databases. DQL commands are typically used to retrieve data from one or more tables in the database. Some examples of DQL commands include SELECT, FROM, WHERE, and ORDER BY.

**Transaction Control Language (TCL)**

TCL is used to control transactions in a database. A transaction is a group of related database operations that are treated as a single unit. TCL commands are used to start, commit, and roll back transactions. Some examples of TCL commands are BEGIN TRANSACTION, COMMIT, and ROLLBACK.

In addition to these five main types of query languages, there are also a number of specialized query languages that are used for specific purposes. For example, there are query languages for querying XML data, graph data, and time series data.

The choice of which query language to use depends on the specific task that you are trying to perform. For example, if you are trying to define the structure of a database, you would use DDL. If you are trying to insert data into a database, you would use DML. And if you are trying to query data from a database, you would use DQL.

1. What is the difference between the main key and a composite key? Give instances of how primary key and composite are used.

Both primary keys and composite keys are used to uniquely identify records in a database table. However, there is a key difference between the two:

Primary Key

A primary key is a single column or a set of columns that uniquely identifies each record in a table. It is a mandatory constraint that cannot contain null values. The primary key is used to enforce data integrity and to ensure that each record in the table is unique.

Composite Key

A composite key is a set of two or more columns that together uniquely identify each record in a table. It is also a mandatory constraint that cannot contain null values. Composite keys are often used when there is no single column that can uniquely identify a record.

Instances of Primary Keys

* The customer ID in a customer table
* The order ID in an order table
* The student ID in a student table

Instances of Composite Keys

* The combination of customer ID and order ID in an order item table
* The combination of student ID and course ID in a student enrollment table
* The combination of employee ID and department ID in an employee table

In general, primary keys are preferred over composite keys because they are simpler and more efficient. However, composite keys are sometimes necessary when there is no single column that can uniquely identify a record.

1. Create a table with a primary key, a column default value, and a column unique constraint in SQL.

here is an example of how to create a table with a primary key, a column default value, and a column unique constraint in SQL:

SQL

CREATE

TABLE customers (

customer\_id INT

NOT

NULL AUTO\_INCREMENT PRIMARY KEY,

first\_name VARCHAR(50) NOT

NULL,

last\_name VARCHAR(50) NOT

NULL,

email VARCHAR(100) UNIQUE,

phone\_number VARCHAR(20) DEFAULT '123-456-7890'

);

Use code with caution. [Learn more](https://bard.google.com/faq#coding)

content\_copy

This code will create a table called customers with six columns:

* customer\_id: This is the primary key of the table. It is an integer column that is automatically incremented for each new record.
* first\_name: This is a varchar column that stores the first name of the customer. It cannot be null.
* last\_name: This is a varchar column that stores the last name of the customer. It cannot be null.
* email: This is a varchar column that stores the email address of the customer. It must be unique.
* phone\_number: This is a varchar column that stores the phone number of the customer. It has a default value of 123-456-7890.